Hi All,

Here is the paper I adapted and implemented for this problem: http://citeseer.ist.psu.edu/226211.html
NOTE: the paper discusses using multiple ant-colonies to optimize multiple objective functions whereas I use only one colony since there is only one objective function (the number of patients saved).

A good general website on the Vehicle Routing Problem (VRP): http://neo.lcc.uma.es/radi-aeb/WebVRP/
A good general description of Ant Colony Optimization (ACO): http://citeseer.ist.psu.edu/396734.html

A quick recap of the idea:
Real live ant colonies discover good routes to food by laying down pheromone trails, which are essentially a collective shared memory. Ants initially wander randomly in search of food, laying down pheromone as they go. If they find food they return, laying down even more pheromone on a good trail. Later ants are more likely to follow a path with more pheromone (thus laying even more pheromone) so over time good paths are reinforced; meanwhile the evaporation of pheromone means bad paths and long paths are more likely to be forgotten.

In our case the ambulance problem scenario is viewed as a complete graph, where the nodes are hospitals and patients. Each ant's task is to explore a *complete* solution to the problem. So the first ant will start from a random ambulance, look at the savable patients, pick one according to a heuristic (see below), look at patients savable from there, pick one, return to the hospital, and so on until the first ambulance cannot save any more patients, then that ant will start from another random ambulance (resetting time to 0 but keeping track of which patients were saved in the runs so far), and so on until all the ambulances have been run. The entire path traced by the ant in this way is a proposed solution to the problem. Then the pheromone trails are updated (see below) and another ant is run, and so on until the stopping criterion (in this case CPU time) is reached. The best scoring solution found is then output.

Updating pheromone:
1. Local updating: after each ant has computed a solution, the edges along the path traced by that ant have their pheromone *reduced*. This is to encourage successive ants to explore different solutions.
2. Global updating: periodically the best solution found so far has

the pheromone along its path increased (reinforcement) and at the same time pheromone is decreased globally along all edges in the graph (evaporation).

Heuristics: picking a patient to save
Almost the entire logic of this algorithm is encoded in how an ant picks the next patient to save among the feasible patients. The logic involves tradeoffs between exploration and exploitation (finding new solutions vs. converging on the best found so far). The following factors are considered:
1. distance to the patient: the farther the patient, the less likely the ant will choose that patient (this favors greediness)
2. remaining time to live: the less time the patient has left to live, the more likely the ant is to choose that patient (this favors saving patients with small windows earlier)
3. pheromone on the path from the current location to the patient (this favors exploration in the neighborhood of the best solution so far)
4. number of times this patient has been saved by previous ants (this favors consideration of patients not yet incorporated in solutions seen so far)

-A